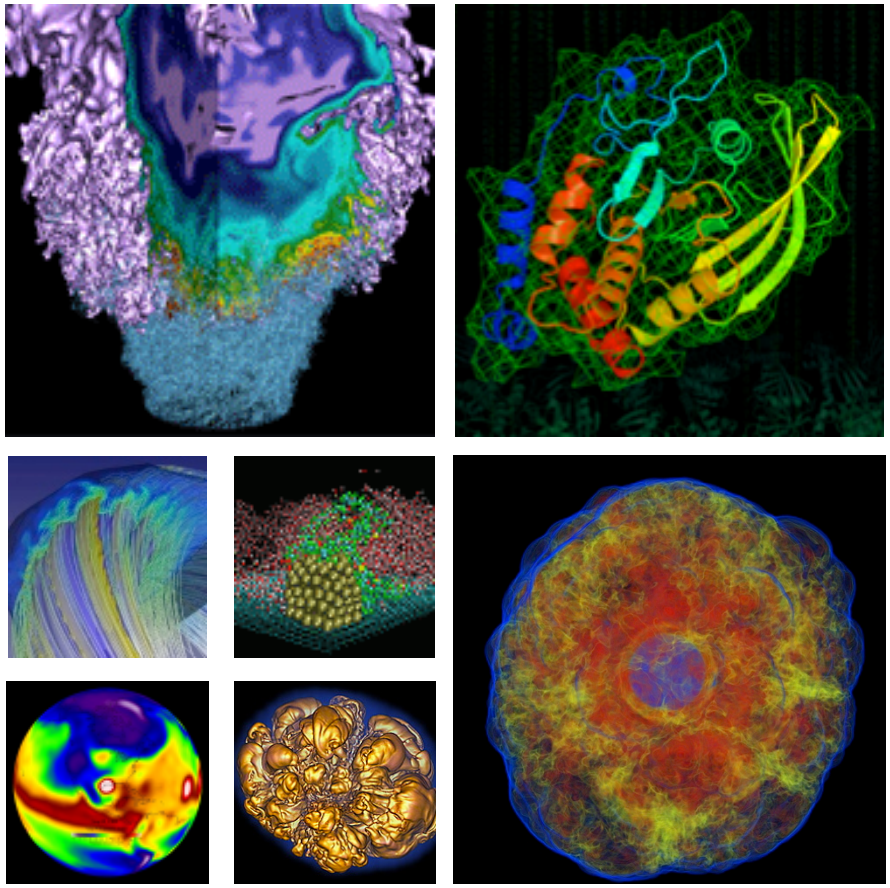
Performance Portability Experiences at NERSC



Brian Friesen, et al.

DOE COE Performance Portability 2017

2017 Aug 23

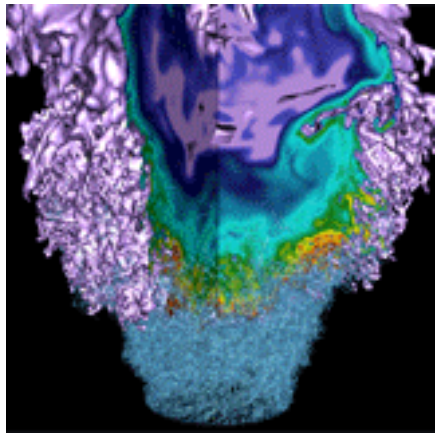
Summary

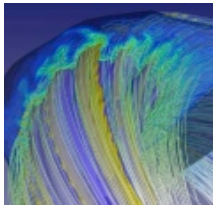
- **We attempted to implement OpenMP 4.x and Kokkos in 3 codes at NERSC:**
  - **BoxLib** (C++/Fortran AMR framework)
  - **BerkeleyGW** (F90 mat. sci. code)
  - **Dslash** (C++ QCD kernel)
- **So far, results have ranged from underwhelming to mixed**
  - Our hands are still full with the "portable" part; have barely touched the "performance" part
- **The goal was to run the same code on both GPUs and KNL (but that was probably too ambitious)**

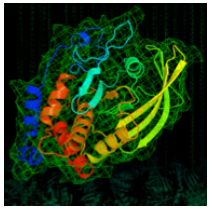
Common Themes

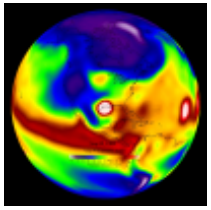
- **OpenMP 4.x: results vary wildly with compiler**
  - Some things crash the compiler (Cray, IBM, PGI)
  - Some things compile but generate the wrong answer (Intel, Cray)
  - Some things compile and run but have bad performance (GCC)
  - Would be nice if OpenMP spec defined the behavior of the *target* construct if no device is available
- **Kokkos: requiring a memory model for a perf.port. framework is OK, unless the existing code already has one (BoxLib)**
  - Then your code "port" can become a complete rewrite

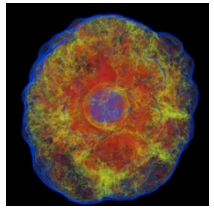
Geometric multigrid solver in BoxLib

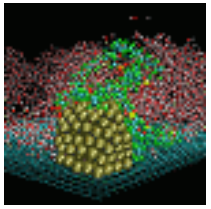












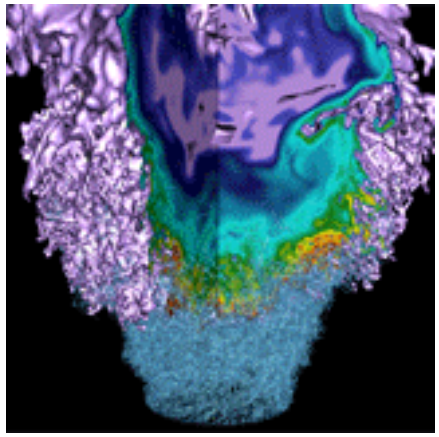
Algorithm overview

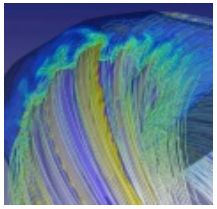
- **Geometric multigrid: an iterative method to solve linear problems on structured grids**
- **C++ framework; calls Fortran kernels to do FLOPs**
- **4 main kernels in GMG:**
  - Restriction − average fine grid onto coarse grid
  - Prolongation − interpolate coarse grid onto fine grid
  - Relaxation − a few iterations of linear solve on a grid
    - E.g., 2 Jacobi iterations, 4 Gauss-Seidel red-black, etc.
  - Bottom solve − exact solution of linear system on coarsest grid
    - Can be a direct method since coarsest grid is small
- **Kernels 1-3 are stencil-ish, the 4th is dense linear algebra**

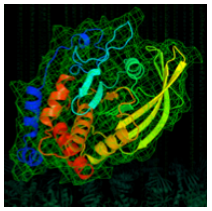
```
    const bool tiling = true;
#ifdef _OPENMP
#pragma omp parallel
#endif
    for (MFIter cmfi(c,tiling); cmfi.isValid(); ++cmfi)
    {
        BL_ASSERT(c.boxArray().get(cmfi.index()) == cmfi.validbox());

        const int         nc   = c.nComp();
        const Box&        bx   = cmfi.tilebox();
        FArrayBox&        cfab = c[cmfi];
        const FArrayBox& ffab = f[cmfi];

        FORT_AVERAGE(cfab.dataPtr(),
                     ARLIM(cfab.loVect()), ARLIM(cfab.hiVect()),
                     ffab.dataPtr(),
                     ARLIM(ffab.loVect()), ARLIM(ffab.hiVect()),
                     bx.loVect(), bx.hiVect(), &nc);
    }
}
```

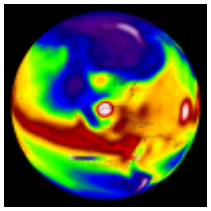
```
      do n = 1, nc
         do k = lo(3), hi(3)
            k2 = 2*k
            k2p1 = k2 + 1
           do j = lo(2), hi(2)
               j2 = 2*j
               j2p1 = j2 + 1
               do i = lo(1), hi(1)
                  i2 = 2*i
                  i2p1 = i2 + 1
                  c(i,j,k,n) =  (
     $                 + f(i2p1,j2p1,k2  ,n) + f(i2,j2p1,k2  ,n)
     $                 + f(i2p1,j2  ,k2  ,n) + f(i2,j2  ,k2  ,n)
     $                 + f(i2p1,j2p1,k2p1,n) + f(i2,j2p1,k2p1,n)
     $                 + f(i2p1,j2  ,k2p1,n) + f(i2,j2  ,k2p1,n)
     $                 )*eighth
               end do
            end do
         end do
      end do
```

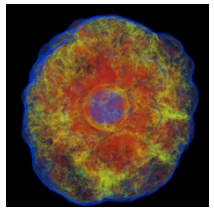
BerkeleyGW kernel

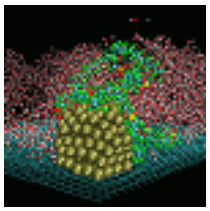








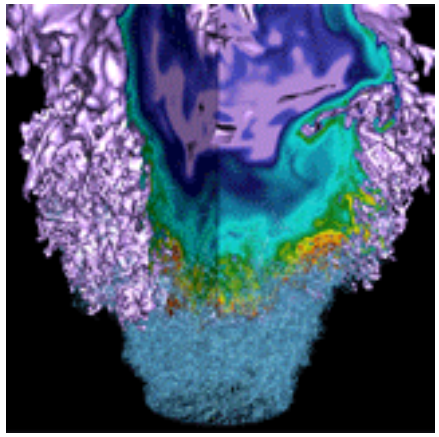


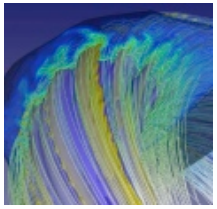


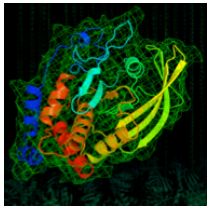
Algorithm overview

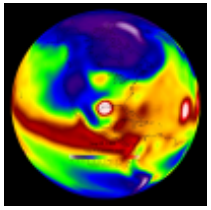
- **F90 MPI+OpenMP mat. sci. code**
  - Predicts excited-state properties of materials
  - Uses GW method (alternative to DFT) – lots of FFTs and dense linear algebra
- **"GPP" kernel from BGW is ~400 LOC kernel in a single file**
- **Written in Fortran, also ported to C++ to test Kokkos**

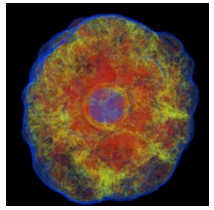
Results so far with Kokkos and OpenMP 4.x

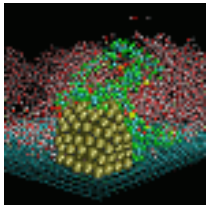












BoxLib + Kokkos

- **BoxLib already has a huge infrastructure of data structures and functions which operate on 2D/3D grids**
  - Computing volume intersections of grids
  - Coarse-fine boundaries on AMR grids
  - Ghost zone exchange
  - Regridding/load balancing
- **Almost none of this was compatible with the Kokkos memory model ("Views") and had to be rewritten**

```
~/BoxLib> git diff --stat cpp_kernels_kokkos-views

 Src/C_BaseLib/FArrayBox.H                           |    2 -
 Src/C_BaseLib/FabArray.H                            |    8 +-
 Src/C_BaseLib/IArrayBox.H                           |    1 -
 Src/C_BaseLib/KArena.H                              |  265 ---------
 Src/C_BaseLib/KBaseFab.H                            | 3615 --------------------------------------------------------
 Src/C_BaseLib/Looping.H                             |  770 +-----------------------
 Src/C_BaseLib/Make.package                          |    6 +-
 Src/C_BaseLib/MultiFabUtil.cpp                      |  284 +++++----
 Src/C_BaseLib/MultiFabUtil_3d.cpp                   |  247 --------
 Src/C_BaseLib/MultiFabUtil_F.H                      |    8 -
 Src/C_BoundaryLib/Mask.H                            |    1 -
 Src/C_BoundaryLib/Mask.cpp                          |    2 +-
 Src/LinearSolvers/C_CellMG/ABecLaplacian.H          |    9 +-
 Src/LinearSolvers/C_CellMG/ABecLaplacian.cpp        | 1119 ++++++++++++++++------------------------
 Src/LinearSolvers/C_CellMG/ABec_3D.F                |    4 +-
 Src/LinearSolvers/C_CellMG/CGSolver.H               |    6 +-
 Src/LinearSolvers/C_CellMG/CGSolver.cpp             |   13 +-
 Src/LinearSolvers/C_CellMG/LO_3D_cpp.cpp            |  235 --------
 Src/LinearSolvers/C_CellMG/LO_F.H                   |    5 -
 Src/LinearSolvers/C_CellMG/Laplacian.H              |    3 +-
 Src/LinearSolvers/C_CellMG/Laplacian.cpp            |  343 ++++++-----
 Src/LinearSolvers/C_CellMG/LinOp.H                  |   12 +-
 Src/LinearSolvers/C_CellMG/LinOp.cpp                |   85 +--
 Src/LinearSolvers/C_CellMG/MG_3D_cpp.cpp            |  464 ---------------
 Src/LinearSolvers/C_CellMG/MG_3D_fortran.F          |   96 ---
 Src/LinearSolvers/C_CellMG/MG_3D_old.cpp            |  222 -------
 Src/LinearSolvers/C_CellMG/MG_F.H                   |   81 ---
 Src/LinearSolvers/C_CellMG/Make.package             |    6 +-
 Src/LinearSolvers/C_CellMG/MultiGrid.H              |    4 +-
 Src/LinearSolvers/C_CellMG/MultiGrid.cpp            | 1463 +++++++++++++++++++++++----------------------
 Src/LinearSolvers/C_CellMG/old/MG_3D_cpp.cpp-average |   39 --
 Src/LinearSolvers/C_CellMG4/ABec2.H                 |    5 +-
 Src/LinearSolvers/C_CellMG4/ABec4.H                 |    3 +-
 Src/LinearSolvers/C_CellMG4/ABec4.cpp               |    7 +-
 Tools/C_mk/Make.rules                               |    4 +-
 Tools/Postprocessing/F_Src/GNUmakefile              |    2 +-
 Tutorials/MultiGrid_C/COEF_3D.F90                   |   14 +-
 Tutorials/MultiGrid_C/COEF_F.H                      |   10 +-
 Tutorials/MultiGrid_C/GNUmakefile                   |   28 +-
 Tutorials/MultiGrid_C/KokkosCore_config.h           |   11 -
 Tutorials/MultiGrid_C/KokkosCore_config.tmp         |   11 -
 Tutorials/MultiGrid_C/MG_helpers_cpp.cpp            |  162 ------
 Tutorials/MultiGrid_C/Make.package                  |    2 +-
 Tutorials/MultiGrid_C/RHS_3D.F90                    |  143 ++---
 Tutorials/MultiGrid_C/RHS_F.H                       |    3 +-
 Tutorials/MultiGrid_C/fcompare                      |  Bin 3475616 -> 0 bytes
 Tutorials/MultiGrid_C/inputs                        |    6 +-
 Tutorials/MultiGrid_C/main.cpp                      | 1530 ++++++++++++++++++++++++-----------------------
 Tutorials/MultiGrid_C/out-F                         |  522 -----------------
 Tutorials/MultiGrid_C/out-cpp                       |  522 -----------------
 55 files changed, 2455 insertions(+), 10764 deletions(-)
```

BerkeleyGW + Kokkos

- No complicated data structures in GPP kernel; implementing Kokkos on hottest loops was straightforward
- (Of course, we had to convert the whole kernel from Fortran to C++ first)

OpenMP in BoxLib and BGW

- **OpenMP does not support reductions over complex numbers in C/C++ (but it does in Fortran)**
- **GCC requires the "simd" construct to parallelize among threads in a threadblock when using "#pragma omp target teams distribute parallel for" (Intel does not; Cray is ??)**
- **Intel requires OMP_NUM_THREADS=(max possible # threads on arch) or else the code segfaults (GCC and Cray do not)**
- **Intel OpenMP 3.x and 4.x give similar performance for "#pragma omp teams distribute parallel for simd schedule(dynamic)", but** . . .
  - If you put in the "simd" statement that GCC needs, then code runs 4x slower

OpenMP in BoxLib and BGW

- **GCC: "target" construct has a major performance bug wherein threads exiting a parallel region are destroyed, not "cached" (GCC bugzilla #80859)**
- **CCE 8.6.0 and 8.6.1 segfault when compiling a BoxLib source file with a "target" construct**
- **Without "target" construct, CCE 8.6.0 and 8.6.1 have link error in BoxLib**
- **IBM: XLC v13.1 fails to link >1 compilation units together if they both include a header file which contains a "target" region**
  - Fixed in v14.0, but now the compiler segfaults

BerkeleyGW + Kokkos

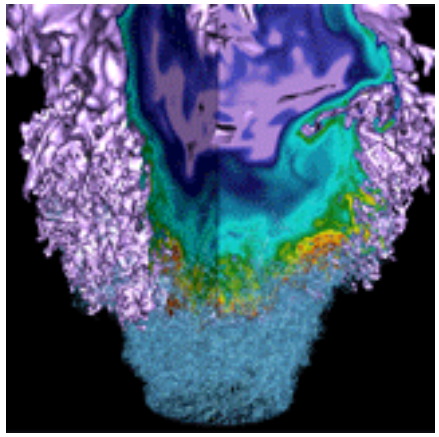
Approach	Architecture	Timings (seconds)
Fortran (Sequential)	KNL	973.5
C++ (Sequential)	KNL	1193.9
Fortran (OpenMP 3.0)	KNL	12.7
C++ (OpenMP 3.0)	KNL	12.8
C++ (OpenMP 4.5)	KNL	16.4
C++ (Kokkos+OpenMP)	KNL	34.2

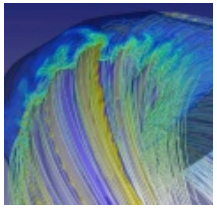
Approach	Architecture	Timings (seconds)
Fortran (Sequential)	PowerPC	935.9
C++ (Sequential)	PowerPC	1263.5
Fortran (OpenMP 3.0)	PowerPC	41
C++ (OpenMP 3.0)	PowerPC	70.1
C++ (Kokkos+OpenMP)	PowerPC	17.03
C++ (Kokkos+CudaUVM)	Pascal	3.93

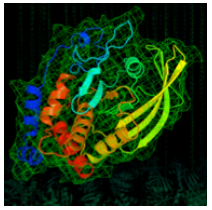
BerkeleyGW + OpenMP

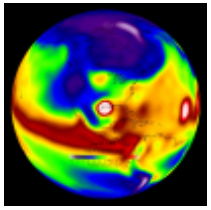
	OpenMP 3.0	OpenMP 4.5
Intel	1.08	3.7 (same even if we add simd)
GCC	12.9	16.6 (13.8 with simd)
Cray	7.08 (non-vectorized)	Too long did not wait for it to end...

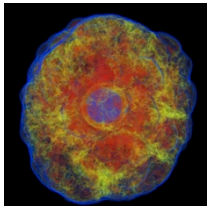
Summary

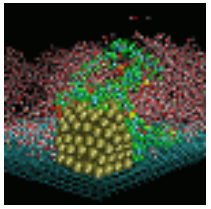












Common Themes

- **OpenMP 4.x: results vary wildly with compiler**
  - Some things crash the compiler (Cray, IBM, PGI)
  - Some things compile but generate the wrong answer (Intel, Cray)
  - Some things compile and run but have bad performance (GCC)
  - Would be nice if OpenMP spec defined the behavior of the *target* construct if no device is available
- **Kokkos: requiring a memory model for a perf.port. framework is OK, unless the existing code already has one (BoxLib)**
  - Then your code "port" can become a complete rewrite

National Energy Research Scientific Computing Center